Aterrizaje vertical de un cohete

Código: CMM917511

2 de octubre de 2023

1. Descenso con mínimo consumo de combustible

Utilizando las notaciones para la discretización de los datos que se proponen en el enunciado, el problema propuesto consiste en calcular los valores de ciertas incógnitas $f_0^x, f_0^y, f_0^z, \cdots, f_{K-1}^x, f_{K-1}^y, f_{K-1}^z$ —correspondientes a las coordenadas (x, y, z) de los perfiles de fuerza f_i— que minimicen la función[1] del consumo de combustible

$$\Gamma(f_0, \cdots, f_{K-1}) = \gamma \Delta t \sum_{i=0}^{k-1} \parallel f_i \parallel_2$$

sujeto a las restricciones de posición final $p_K = 0$, velocidad final $v_K = 0$, las restricciones de las posiciones intermedias $p_i^z \geq \alpha \parallel (p_i^x, p_i^y) \parallel_2$ y la restricción de fuerza máxima $\parallel f_i \parallel_2 \leq F^{max}$.

1.1. Modelización del problema

Antes de presentar un modelo de optimización con el que podamos trabajar, es necesario que todos los elementos que van a intervenir en el mismo —función objetivo, restricciones, etc.— sean dependientes exclusivamente de las variables de decisión, que en este caso serán las componentes de los perfiles de fuerza. Por tanto, lo primero será desarrollar una expresión general —no recursiva— de las velocidades v_i y las posiciones p_i que dependan exclusivamente de los perfiles de fuerza f_i, pues ambas magnitudes intervienen en las restricciones comentadas al inicio. De esta manera, la expresión general para ambas —cuya cuyas demostraciones A.1 y A.2 se encuentran en el anexo final— es la siguiente:

$$v_j = v_0 - j\Delta t g\vec{e_3} + \frac{\Delta t}{m} \sum_{n=0}^{j-1} f_n \tag{1}$$

$$p_j = p_0 + \Delta t j v_0 - \Delta t^2 \cdot \frac{j^2}{2} g\vec{e} + \frac{\Delta t^2}{m} \sum_{n=0}^{j-1} \left(j - n - \frac{1}{2} \right) f_n \tag{2}$$

Las ecuaciones 1 y 2 particularizadas para la posición final p_K y la velocidad final v_K permiten reescribir las restricciones en las que intervenían de la siguiente manera:

$$\begin{cases} v_K = 0 \\ p_K = 0 \end{cases} \Rightarrow \begin{cases} \frac{\Delta t}{m} \sum_{n=0}^{K-1} f_n = -v_0 + K\Delta t g\vec{e_3} \\ \frac{\Delta t^2}{m} \sum_{n=0}^{K-1} \left(K - n - \frac{1}{2} \right) f_n = -p_0 - \Delta t K v_0 + \Delta t^2 g \frac{K^2}{2} \vec{e} \end{cases}$$

En consecuencia, si desarrollamos el sistema anterior en las tres componentes de cada vector, éste queda como:

$$\begin{pmatrix} \frac{\Delta t}{m} & 0 & 0 & & \frac{\Delta t}{m} & 0 & 0 \\ 0 & \frac{\Delta t}{m} & 0 & \cdots & 0 & \frac{\Delta t}{m} & 0 \\ 0 & 0 & \frac{\Delta t}{m} & & 0 & 0 & \frac{\Delta t}{m} \\ \frac{\Delta t^2 \left(K - \frac{1}{2}\right)}{m} & 0 & 0 & & \frac{\Delta t^2 \left(1 - \frac{1}{2}\right)}{m} & 0 & 0 \\ 0 & \frac{\Delta t^2 \left(K - \frac{1}{2}\right)}{m} & 0 & \cdots & 0 & \frac{\Delta t^2 \left(1 - \frac{1}{2}\right)}{m} & 0 \\ 0 & 0 & \frac{\Delta t^2 \left(K - \frac{1}{2}\right)}{m} & & 0 & 0 & \frac{\Delta t^2 \left(1 - \frac{1}{2}\right)}{m} \end{pmatrix} \begin{pmatrix} f_0^x \\ f_0^y \\ f_0^z \\ \vdots \\ f_{K-1}^x \\ f_{K-1}^y \\ f_{K-1}^z \end{pmatrix} = \begin{pmatrix} -v_0^x \\ -v_0^y \\ -v_0^z + K\Delta t g \\ -p_0^x - \Delta t K v_0^x \\ -p_0^y - \Delta t K v_0^y \\ -p_0^z - \Delta t K v_0^z + \Delta t^2 g \frac{K^2}{2} \end{pmatrix}$$

[1] La aplicación se ha escrito en función de los valores de los perfiles f_i directamente por simplicidad visual, pero en realidad habría que escribir todas las componentes de cada perfil como variables de la función.

donde la matriz de coeficientes —que denotaremos A_{eq}— es de tamaño $6 \times 3K$ y la matriz de términos independientes —que denotaremos b_{eq}— es de tamaño 6×1. De esta manera, tenemos un sistema lineal indeterminado con $3K - 6$ variables independientes y 6 dependientes.

De nuevo, gracias a las expresiones de las ecuaciones 1 y 2, las restricciones no lineales sobre las posiciones durante el aterrizaje

$$(p_i^z)^2 - \alpha^2 (p_i^x)^2 - \alpha^2 (p_i^y)^2 \geq 0$$

quedarían como:

$$\left(\overbrace{p_0^z + i\Delta t v_0^z - \Delta t^2 \frac{i^2}{g}}^{A_1} + \overbrace{\Delta t^2 \frac{1}{m} \sum_{j=0}^{i-1} \left(i - j - \frac{1}{2} \right) f_j^z}^{A_2} \right)^2$$
$$-\alpha^2 \left(\overbrace{p_0^x + i\Delta t v_0^x}^{B_1} + \overbrace{\Delta t^2 \frac{1}{m} \sum_{j=0}^{i-1} \left(i - j - \frac{1}{2} \right) f_j^x}^{B_2} \right)^2$$
$$-\alpha^2 \left(\overbrace{p_0^y + i\Delta t v_0^y}^{C_1} + \overbrace{\Delta t^2 \frac{1}{m} \sum_{j=0}^{i-1} \left(i - j - \frac{1}{2} \right) f_j^y}^{C_2} \right)^2 \geq 0$$

y utilizando la notación que indicada para simplificar la escritura, el desarrollo de cuadrados de las desigualdades resultaría en:

$$-A_2^2 - 2A_1A_2 + \alpha^2 B_2^2 + \alpha^2 2B_1B_2 + \alpha^2 C_2^2 + \alpha^2 2C_1C_2 \leq A_1^2 - \alpha^2 B_1^2 - \alpha^2 C_1^2$$

Una vez reescritas las restricciones del problema, estamos en condiciones de poder escribir el modelo matemático de optimización que describe el problema propuesto:

El problema $\mathcal{P}$ se plantea como:

- Variable de decisión:
$$f := (f_0^x, f_0^y, f_0^z, \cdots, f_{K-1}^x, f_{K-1}^y, f_{K-1}^z)$$
- Función objetivo:
$$\Gamma : \mathbb{R}^{3K} \longrightarrow \mathbb{R}$$
$$(f_0, \cdots, f_K) \longmapsto \gamma \Delta t \sum_{i=0}^{K-1} \parallel f_i \parallel_2$$
- Restricciones lineales:
$$A_{eq} \cdot f = b_{eq}$$
- Restricciones no lineales
$$\forall i \in \{1, \cdots, K\} : -A_2^2 - 2A_1A_2 + \alpha^2 B_2^2 + \alpha^2 2B_1B_2 + \alpha^2 C_2^2 + \alpha^2 2C_1C_2 \leq A_1^2 - \alpha^2 B_1^2 - \alpha^2 C_1^2$$
$$\forall i \in \{0, \cdots, K-1\} : \parallel f_i \parallel_2 \leq F^{max}$$

Figura 1: Modelo completo de optimización del problema

La modelización que se presenta es un problema de optimización no lineal que se resuelve con técnicas de programación no lineal, como el *Metodo de Newton* o el de *Multiplicadores de Lagrange*. Para ello, hemos empleado la función `fmincon` de *MatLab* que emplea distintos algoritmos en función de la entrada propuesta.

En cualquier caso, hay un elemento imprescindible en el uso de cualquiera de estos algoritmos: la obtención de una solución inicial factible —no necesariamente óptima—. Sin embargo, esto representa un problema grave a

la hora de resolver el problema abordándolo por medio de estas técnicas, pues en general una solución factible no es sencilla de encontrar y para K elevado la heurísitica no es una opción.

1.2. Cálculo de soluciones iniciales factibles

Para encontrar dicha solución inicial factible, hemos readaptado una de las técnicas de programación lineal para la inicialización del algoritmo del Símplex conocida como *el método de las dos fases*. El procedimiento consiste en elaborar un nuevo problema $\mathcal{P}'$ a partir del problema $\mathcal{P}$ original a través de los siguientes pasos:

1. Incluir variables "artificiales" x_i^a como un sumando más en todas[2] las restricciones.
2. Cambiar la función objetivo a la función suma de todas las variables artificiales.
3. Introducir la restricción de que todas las variables artificiales sean mayores o iguales que 0.

Así pues, resolver $\mathcal{P}'$ consiste en minimizar la suma de las variables artificiales. Como todas son mayores o iguales a cero, la resolución consiste en "anular" las nuevas variables artificiales. Si se consigue una solución óptima de $\mathcal{P}'$ que anule todas las variables artificiales, se habrá conseguido una tupla de variables de decisión del problema original que verifica las restricciones del problema original (aunque no sea la óptima), mientras que si no se consigue una solución óptima que anule todas las variables artificiales no habrá solución factible del problema $\mathcal{P}$ original.

Ahora, se puede obtener una solucion inicial factible para $\mathcal{P}'$ de manera sencilla anulando las variables de decisión y escogiendo las variables artificiales de manera que cumplan las restricciones. Observamos que tal y como tenemos escritas las restricciones tienen una de las siguientes formas: $\Theta(f) = \Omega$ o $\Theta(f) \leq \Omega$. Observamos que el término de la izquierda depende de f y se anula al anularse f mientras que el término de derecha es independiente de f. Por tanto, dada una restricción, la variable artificial debe introducirse en el lado izquierdo sumando, si el término de la izquierda es positivo, o restando, si el término i es negativo.

$$\begin{cases} \Theta(f) + x_a = \Omega \text{ o } \Theta(f) + x_a \leq \Omega & \text{si } \Omega >= 0 \\ \Theta(f) - x_a = \Omega \text{ o } \Theta(f) - x_a \leq \Omega & \text{si } \Omega < 0 \end{cases}$$

De este modo, al asignar a la variable artificial el valor $|\Omega|$ y al anular f se cumple la restriccion. Hemos obtenido una solución factible el problema $\mathcal{P}'$.

Observamos además que en las restricciones que signo $\leq$ con $\Omega > 0$ no hace falta introducir variable artificial ya que simplemente anulando f se cumple la restricción. Haremos esto en la implementación.

Una vez calculada la solución inicial a través del procedimiento descrito, podemos emplear la misma para resolver el problema $\mathcal{P}$ original.

1.3. Resumen apartado (a)

Así pues la resolucón del apartado (a) consiste en estas dos fases.

- **fase 1 (Inicialización)**: Se transforma el problema $\mathcal{P}$ en el problema $\mathcal{P}'$ y se resuelve el problema $\mathcal{P}'$ para encontrar una solución inicial factible del problema $\mathcal{P}$.
- **fase 2 (Resolución)**: se utiliza la solución inicial factible obtenida en la fase anterior para resolver el problema $\mathcal{P}$.

Los detalles de la implementación se exponerán en un apartado dedicado posterior.

2. Tiempo mínimo de descenso

Para obtener el valor de K mínimo tal que el problema $\mathcal{P}$ original sea factible, partimos de un valor de K para el cual sabemos que es factible y realizamos una *búsqueda binaria* en el espacio de soluciones $[1, \cdots, K]$. En

[2] En rigor es en todas, pero para simplificar el problema computacionalmente no se han incluido en aquellas restricciones que se verifiquen trivialmente si todas las variables de decisión se anulan.

cada paso dividimos el espacio por la mitad y comprobamos si para el valor $mathcal(K)$ intermedio el problema es factible usando el algoritmo de la primera fase. Si no lo es debemos buscar en la mitad superior del espacio restante pues el aterrizaje requiere de más tiempo, mientras que si sí lo es podemos reducir el tiempo y buscamos en la mitad inferior del espacio restante.

Los detalles de la implementación se ecuentran posteriormente en una sección dedicada.

3. Funcionamiento del código de MatLab y resultados

La ejecución de los archivos `apartado_a.m` y `apartado_b.m` ejecutan el funcionamiento correspondiente a ambos apartados.

Para el apartado (a) se sigue el siguiente orden de ejecución:

1. Cálculo de solución inicial:

   La sentencia

   `[sol_inicial, fval, exitflag, output] = inicializacion(K, p_0, v_0, p_K, v_K, delta_t, m, g, Fmax, alpha, options);`

   nos proporciona una solución inicial para el problema de optimización en la variable `sol_inicial`.

   Dentro de la función `inicializacion` se hace referencia a funciones para calcular las matrices de las restricciones lineales y las funciones de las restricciones no lineales.

   Con `init_va_no_lineal = inicializacion_aritificial_no_lineal(K, p_0, v_0, delta_t, g, alpha);` obtenemos una solución inicial donde las únicas variables no nulas son las artificiales.

   Tras esto hacemos la llamada a `fmincon`, utilizando como función objetivo la suma de las variables artificiales (todas positivas, por tanto la suma también), que queremos que valga 0.

   Aunque no se haga uso de esta función en la ejecución final, durante el proceso de desarrollo hicimos uso de las funciones `comprueba_cono` (en referencia a la forma que tiene la restricción sobre los puntos) y `comprueba_f` para comprobar que las soluciones obtenidas en esta fase y en la siguiente se ajustaban a las restricciones del enunciado. Las funciones `calcula_v_y_p` y `de_p_a_xyz` facilitan el uso de la función `comprueba_cono`;

2. Cálculo del uso de combustible mínimo

   La llamada `[f,fval, exitflag, output] = optimizacion(sol_inicial, K, p_0, v_0, p_K, v_K, delta_t, m, g, Fmax, alpha, gamma, options);` desencadena la resolución del problema de optimización para el cálculo del uso mínimo de combustible. El funcionamiento de esta función es muy similar al de la función previamente explicada. El cambio principal es la función objetivo, que calcula la suma de las normas euclídeas de las fuerzas, todo ello multiplicado por $\gamma \cdot \Delta t$. Se puede observar en el código que hay una opción para mayor precisión, repitiendo la llamada a `fmincon` varias veces. Esto se debe a que, debido a las restricciones de iteraciones, evaluaciones y otras (que se especifican en el argumento `options`) la solución que se obtiene es muy próxima a la óptima, pero no exacta. Repitiendo el proceso varias veces, usando como solución inicial al obtenida previamente, el resultado se aproxima cada vez más al óptimo. No obstante, tras experimentación y pruebas vimos que generalmente el resultado se estabiliza a partir de cinco iteraciones.

   Tras ello, el resultado óptimo encontrado para el consumo de combustible es aproximadamente 187 (en el caso de mayor precisión que llevamos a cabo se obtuvo 186,89). Hemos dejado un valor menor de iteraciones en el código para una ejecución más breve.

   Para obtener la mayor precisión en caso de quererse se pueden modificar el número de repeticiones de `fmincon` en `optimizacion` y modificar los valores del parámetro `options` para hacerlos más estrictos. De la misma manera, para obtener tiempos de ejecución menores se pueden modificar los valores de `options` para hacerlos más suaves.

   La siguiente línea permite una precisión muy buena:

   `options = optimoptions('fmincon', 'MaxIterations', 10000, 'MaxFunctionEvaluations', 60000, 'StepTolerance',1e-15);`

3. Representación gráfica de la solución:

La última parte de la función `apartado_a` es la llamada a `representacion_trayectoria(f, K, delta_t, g, m);`. Todos los detalles necesarios se pueden consultar en el propio código.

Para ver esta representación sin ejecutar el código completo puede consultarse el archivo `Trayectoria.gif`.

Para el apartado (b) la ejecución es más sencilla:

A sabiendas de que $K = 35$ nos lleva a una solución válida, y que trivialmente $K = 0$ no, para buscar el K mínimo que permite que el cohete aterrice utilizamos dichos valores como cotas superior e inferior respectivamente para hacer búsqueda binaria, funcionalidad llevada a cabo en `busca_K_minimo`. En cada etapa de la búsqueda se invoca la función `inicialización` para que calcule una solución inicial al problema (que de existir, es suficiente, pues no buscamos minimizar uso de combustible en este apartado). No obstante, por el uso del método de las dos fases, solo podemos considerar la solución válida si la suma de las variables artificiales es 0. Debido a posibles errores de precisión leves, la condición de aceptación de la solución calculada es que todas las variables artificiales tengan un valor menos a 10^{-5}. Se puede ver esto implementado en la función de `busca_K_minimo`. Al igual que en el apartado anterior, se pueden modificar los niveles de precisión con el parámetro `options`.

Tras múltiples pruebas con diversos valores de precisión y tiempos de ejecución, el resultado obtenido es que el menor valor de K que permite el aterrizaje bajo las condiciones impuestas es el propuesto por el enunciado, $K = 35$. Por tanto, el tiempo mínimo de descenso es 35.

A. Demostración de resultados mencionados

Proposición A.1. *La fórmula recursiva de la velocidad, definida como*

$$v_{j+1} = v_j + \frac{\Delta t}{m} f_j - \Delta t g \vec{e_3}$$

puede expresarse como la fórmula de de término general

$$v_j = v_0 - j\Delta t g \vec{e_3} + \frac{\Delta t}{m} \sum_{n=0}^{j-1} f_n$$

Demostración. Primero, desplegamos la expresión para formular la hipótesis de inducción. En el desarrollo que se muestra a continuación, el valor $n := j$ deja el despliegue en términos del valor v_0, que es el caso base de la recursión y no permite desplegar recursivamente más la expresión.

$$\begin{aligned} v_j &= v_{j-1} + \frac{\Delta t}{m} f_{j-1} - \Delta t g \vec{e_3} \\ &= v_{j-2} + \frac{\Delta t}{m} f_{j-2} + \frac{\Delta t}{m} f_{j-1} - 2\Delta t g \vec{e_3} \\ &= \cdots \\ &= v_{j-n} + \frac{\Delta t}{m} \sum_{i=0}^{n-1} f_i - n\Delta t g \vec{e_3} \end{aligned}$$

Ahora, para probar la hipótesis, basta con probar dicha fórmula por inducción, comprobando que se verifica para[3] v_1 la hipótesis

$$v_1 = v_0 + \frac{\Delta t}{m} f_0 - \Delta t g \vec{e} = v_0 - 1 \cdot \Delta t g \vec{e} + \frac{\Delta t}{m} \sum_{i=0}^{0} f_i$$

y que suponiéndola probada para j, podemos probarla para $j + 1$, es decir:

$$\begin{aligned} v_{j+1} &= v_j + \frac{\Delta t}{m} f_j - \Delta t g \vec{e_3} \\ &= \left(v_0 - j\Delta t g \vec{e_3} + \frac{\Delta t}{m} \sum_{n=0}^{j-1} f_n \right) + \frac{\Delta t}{m} f_j - \Delta t g \vec{e_3} \\ &= v_0 - (j+1)\Delta t g \vec{e_3} + \frac{\Delta t}{m} \sum_{n=0}^{j} f_n \end{aligned}$$

[3] También se podría haber dicho que trivialmente estaba probado para v_0, sin embargo, se ha escogido v_1 para explicitar que no se verifica sólo para el caso base de la recursión.

Por tanto, por inducción, queda probada la fórmula general. □

Proposición A.2. *La fórmula recursiva de la posición, definida como*

$$p_{j+1} = p_j + \frac{\Delta t}{2}\left(v_j + v_{j+1}\right)$$

puede expresarse como la fórmula de de término general

$$p_j = p_0 + \Delta t j v_0 - \Delta t^2 \cdot \frac{j^2}{2} g\vec{e} + \frac{\Delta t^2}{m} \sum_{n=0}^{j-1} \left(j - n - \frac{1}{2}\right) f_n$$

Demostración. Primero, desplegamos la expresión para formular la hipótesis de inducción. En el desarrollo que se muestra a continuación, el valor $n := j$ deja el despliegue en términos del valor p_0, que es el caso base de la recursión y no permite desplegar recursivamente más la expresión.

$$\begin{aligned}
p_j &= p_{j-1} + \Delta t \left(\frac{1}{2} v_{j-1} + \frac{1}{2} v_j\right) \\
&= p_{j-2} + \Delta t \left(\frac{1}{2} v_{j-2} + v_{j-1} + \frac{1}{2} v_j\right) \\
&= p_{j-n} + \Delta t \left(\frac{1}{2} v_{j-n} + \sum_{i=j-n+1}^{j-1} v_i + \frac{1}{2} v_j\right)
\end{aligned}$$

Sin embargo, como hemos calculado el término general de las v_i, podemos sustituir la expresión general en la fórmula anterior, obteniendo la expresión

$$\begin{aligned}
p_j &= p_0 + \Delta t \left(\frac{1}{2} v_0 + \sum_{i=1}^{j-1} v_i + \frac{1}{2} v_j\right) \\
&= p_0 + \Delta t \left(\frac{1}{2} v_0 + \sum_{i=1}^{j-1} \left(v_0 + \frac{\Delta t}{m} \sum_{n=0}^{i-1} f_n - i \Delta t g \vec{e_3}\right) + \frac{1}{2}\left(v_0 + \frac{\Delta t}{m} \sum_{i=0}^{j-1} f_i - j \Delta t g \vec{e_3}\right)\right) \\
&= p_0 + \Delta t j v_0 - \Delta t^2 g \frac{j^2}{2} \vec{e_3} + \frac{\Delta t^2}{m} \sum_{i=1}^{j-1} \sum_{n=0}^{i-1} f_n + \frac{1}{2} \frac{\Delta t^2}{m} \sum_{i=0}^{j-1} f_i \\
&= p_0 + \Delta t j v_0 - \Delta t^2 g \frac{j^2}{2} \vec{e_3} + \frac{\Delta t^2}{m} \sum_{i=0}^{j-1} (j - 1 - i) f_i + \frac{1}{2} \frac{\Delta t^2}{m} \sum_{i=0}^{j-1} f_i \\
&= p_0 + \Delta t j v_0 - \Delta t^2 g \frac{j^2}{2} \vec{e_3} + \frac{\Delta t^2}{m} \sum_{i=0}^{j-1} \left(j - i - \frac{1}{2}\right) f_i
\end{aligned}$$

Ahora, para probar la hipótesis, basta con probar dicha fórmula por inducción, comprobando que se verifica para[4] p_1 la hipótesis

$$\begin{aligned}
p_1 &= p_0 + \Delta t \left(\frac{1}{2} v_0 + \frac{1}{2} v_1\right) \\
&= p_0 + \Delta t \left(\frac{1}{2} v_0 + \sum_{i=1}^{0} v_i + \frac{1}{2} v_1\right) \\
&= p_0 + \Delta t \left(\frac{1}{2} v_0 + \frac{1}{2}\left(v_0 + \frac{\Delta t}{m} \sum_{i=0}^{0} f_i - \Delta t g \vec{e}\right)\right) \\
&= p_0 + \Delta t v_0 + \Delta t^2 \frac{1}{2} \left(v_0 + \frac{\Delta t}{m} \sum_{i=0}^{0} f_i - \Delta t g \vec{e}\right)
\end{aligned}$$

[4] También se podría haber dicho que trivialmente estaba probado para p_0, sin embargo, se ha escogido p_1 para explicitar que no se verifica sólo para el caso base de la recursión.

y se verifica por la igualdad probada[5] anteriormente al sustituir la expresión general de las v_i. Ahora, suponiéndola probada para j, probémosla para $j+1$, es decir:

$$\begin{aligned} p_{j+1} &= p_j + \frac{\Delta t}{2}\left(v_j + v_{j+1}\right) \\ &\overset{\text{HI}}{=} \left(p_0 + \Delta t j v_0 - \Delta t^2 g \frac{j^2}{2}\vec{e_3} + \frac{\Delta t^2}{m}\sum_{i=0}^{j-1}\left(j - i - \frac{1}{2}\right) f_i\right) + \frac{\Delta t}{2}\left(v_j + v_{j+1}\right) \end{aligned}$$

Ahora, sustituimos en v_j y en v_{j+1} las expresiones dadas por la formula general de las velocidades calculada anteriormente.

$$\begin{aligned} p_{j+1} &= p_0 + \Delta t j v_0 - \Delta t^2 g \frac{j^2}{2}\vec{e_3} + \frac{\Delta t^2}{m}\sum_{i=0}^{j-1}\left(j - i - \frac{1}{2}\right) f_i \\ &\quad + \frac{\Delta t}{2}\left(v_0 - j\Delta t g\vec{e_3} + \frac{\Delta t}{m}\sum_{i=0}^{j-1} f_i + v_0 - (j+1)\Delta t g\vec{e_3} + \frac{\Delta t}{m}\sum_{i=0}^{j} f_i\right) \\ &= p_0 + \Delta t(j+1)v_0 - \Delta t^2 g\frac{j^2}{2}\vec{e_3} - \Delta t^2 g\frac{j}{2}\vec{e_3} - \Delta t^2 g\frac{j+1}{2}\vec{e_3} + \frac{\Delta t^2}{m}\left(\sum_{i=0}^{j-1}\left(j - i - \frac{1}{2}\right) f_i + \frac{1}{2}\sum_{i=0}^{j-1} f_i + \frac{1}{2}\sum_{i=0}^{j} f_i\right) \\ &= p_0 + \Delta t(j+1)v_0 - \Delta t^2 g\frac{j^2 + 2j + 1}{2}\vec{e_3} + \frac{\Delta t^2}{m}\left(\sum_{i=0}^{j-1}\left(j + 1 - i - \frac{1}{2}\right) f_i + \frac{1}{2} f_j\right) \\ &= p_0 + \Delta t(j+1)v_0 - \Delta t^2 g\frac{(j+1)^2}{2}\vec{e_3} + \frac{\Delta t^2}{m}\left(\sum_{i=0}^{j}\left(j + 1 - i - \frac{1}{2}\right) f_i\right) \end{aligned}$$

Por tanto, por inducción, queda probada la fórmula general. □

[5] Nótese que dicha igualdad lo que ha probado explícitamente es que se puede pasar desde la expresión desplegada hasta la que conforma nuestra hipótesis de inducción, sin más que sustituir por el j correspondiente (sin necesidad de probar que la hipótesis es, en efecto, cierta).